Importing Libraries

In [118]:
```
import pandas as pd
import numpy as np
import os
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [119]:
```
# Problem 2:
```

Loading the data set

In [120]:
```
df_clg = pd.read_csv('Education+-+Post+12th+Standard.csv')
```

Basic Data Exploration

In this step, we will perform the below operations to check what the data set comprises of. We will check the below things:

- **head of the dataset**
- **shape of the dataset**
- **info of the dataset**
- **summary of the dataset**

In [121]:
```
df_clg.head().T
```

Out[121]:

	0	1	2	3	4
Names	Abilene Christian University	Adelphi University	Adrian College	Agnes Scott College	Alaska Pacific University
Apps	1660	2186	1428	417	193
Accept	1232	1924	1097	349	146
Enroll	721	512	336	137	55
Top10perc	23	16	22	60	16
Top25perc	52	29	50	89	44
F.Undergrad	2885	2683	1036	510	249
P.Undergrad	537	1227	99	63	869
Outstate	7440	12280	11250	12960	7560
Room.Board	3300	6450	3750	5450	4120
Books	450	750	400	450	800
Personal	2200	1500	1165	875	1500
PhD	70	29	53	92	76
Terminal	78	30	66	97	72
S.F.Ratio	18.1	12.2	12.9	7.7	11.9
perc.alumni	12	16	30	37	2
Expend	7041	10527	8735	19016	10922
Grad.Rate	60	56	54	59	15

head function will tell you the top records in the data set. By default python shows you only top 5 records.

In [122]:
```
df_clg.shape
```

Out[122]:
```
(777, 18)
```

Shape attribute tells us number of observations and variables we have in the data set. It is used to check the dimension of data. The College data set has 777 observations and 18 variables in the data set.

In [123]:
```
df_clg.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 777 entries, 0 to 776
Data columns (total 18 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Names        777 non-null    object 
 1   Apps         777 non-null    int64  
 2   Accept       777 non-null    int64  
 3   Enroll       777 non-null    int64  
 4   Top10perc    777 non-null    int64  
 5   Top25perc    777 non-null    int64  
 6   F.Undergrad  777 non-null    int64  
 7   P.Undergrad  777 non-null    int64  
 8   Outstate     777 non-null    int64  
 9   Room.Board   777 non-null    int64  
 10  Books        777 non-null    int64  
 11  Personal     777 non-null    int64  
 12  PhD          777 non-null    int64  
 13  Terminal     777 non-null    int64  
 14  S.F.Ratio    777 non-null    float64
 15  perc.alumni  777 non-null    int64  
 16  Expend       777 non-null    int64  
 17  Grad.Rate    777 non-null    int64  
dtypes: float64(1), int64(16), object(1)
memory usage: 109.4+ KB
```

In [124]:
```
df_clg.describe().T
```

Out[124]:

	count	mean	std	min	25%	50%	75%	max
Apps	777.0	3001.638353	3870.201484	81.0	776.0	1558.0	3624.0	48094.0
Accept	777.0	2018.804376	2451.113971	72.0	604.0	1110.0	2424.0	26330.0
Enroll	777.0	779.972973	929.176190	35.0	242.0	434.0	902.0	6392.0
Top10perc	777.0	27.558559	17.640364	1.0	15.0	23.0	35.0	96.0
Top25perc	777.0	55.796654	19.804778	9.0	41.0	54.0	69.0	100.0
F.Undergrad	777.0	3699.907336	4850.420531	139.0	992.0	1707.0	4005.0	31643.0
P.Undergrad	777.0	855.298584	1522.431887	1.0	95.0	353.0	967.0	21836.0
Outstate	777.0	10440.669241	4023.016484	2340.0	7320.0	9990.0	12925.0	21700.0
Room.Board	777.0	4357.526384	1096.696416	1780.0	3597.0	4200.0	5050.0	8124.0
Books	777.0	549.380952	165.105360	96.0	470.0	500.0	600.0	2340.0
Personal	777.0	1340.642214	677.071454	250.0	850.0	1200.0	1700.0	6800.0
PhD	777.0	72.660232	16.328155	8.0	62.0	75.0	85.0	103.0
Terminal	777.0	79.702703	14.722359	24.0	71.0	82.0	92.0	100.0
S.F.Ratio	777.0	14.089704	3.958349	2.5	11.5	13.6	16.5	39.8
perc.alumni	777.0	22.743887	12.391801	0.0	13.0	21.0	31.0	64.0
Expend	777.0	9660.171171	5221.768440	3186.0	6751.0	8377.0	10830.0	56233.0
Grad.Rate	777.0	65.463320	17.177710	10.0	53.0	65.0	78.0	118.0

Check for NA / Duplicate records

In [125]:
```
df_clg.isna().sum().sum()
```

Out[125]: 0

In [126]:
```
# Check for duplicate data
dups = df_clg.duplicated()
dups.sum()
```

Out[126]: 0

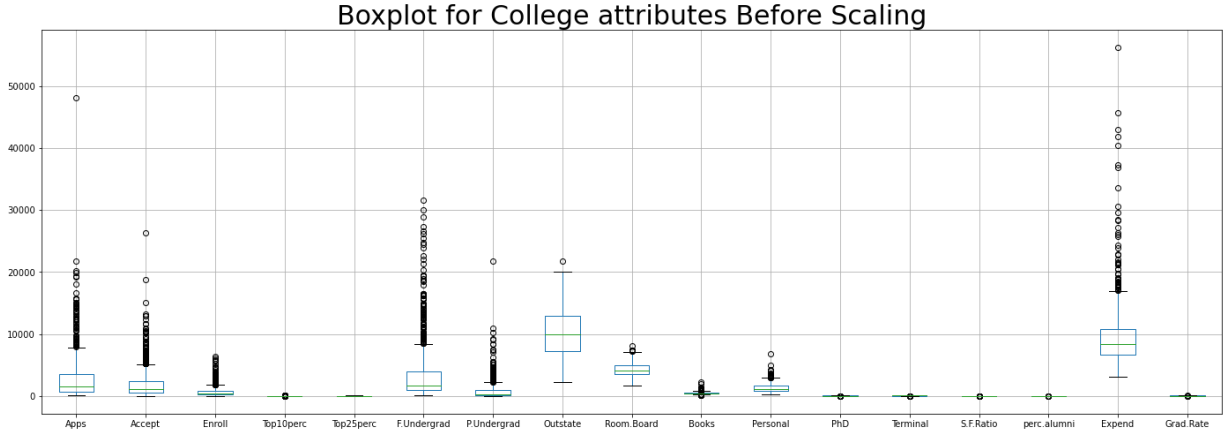
EDA: Univariate Analysis

In [127]:
```
def univariateAnalysis_numeric(column,nbins):
    print("Description of " + column)
    print("----------------------------------------------------------------------------")
    print(df_clg[column].describe(),end=' ')


    plt.figure()
    print("Distribution of " + column)
    print("----------------------------------------------------------------------------")
    sns.histplot(df_clg[column], kde=False, color='g');
    plt.show()

    plt.figure()
    print("BoxPlot of " + column)
    print("----------------------------------------------------------------------------")
    ax = sns.boxplot(x=df_clg[column])
    plt.show()
```

In [128]:
```
df_num = df_clg.select_dtypes(include= ['float64','int64'])
listNumericColumns = list(df_num.columns.values)
len(listNumericColumns)
```

Out[128]: 17

In [129]:
```
df_num.boxplot(figsize=(24,8));
plt.title('Boxplot for College attributes Before Scaling', fontsize=30)
plt.show()
```



In [130]:
```
df_num.head()
```

Out[130]:

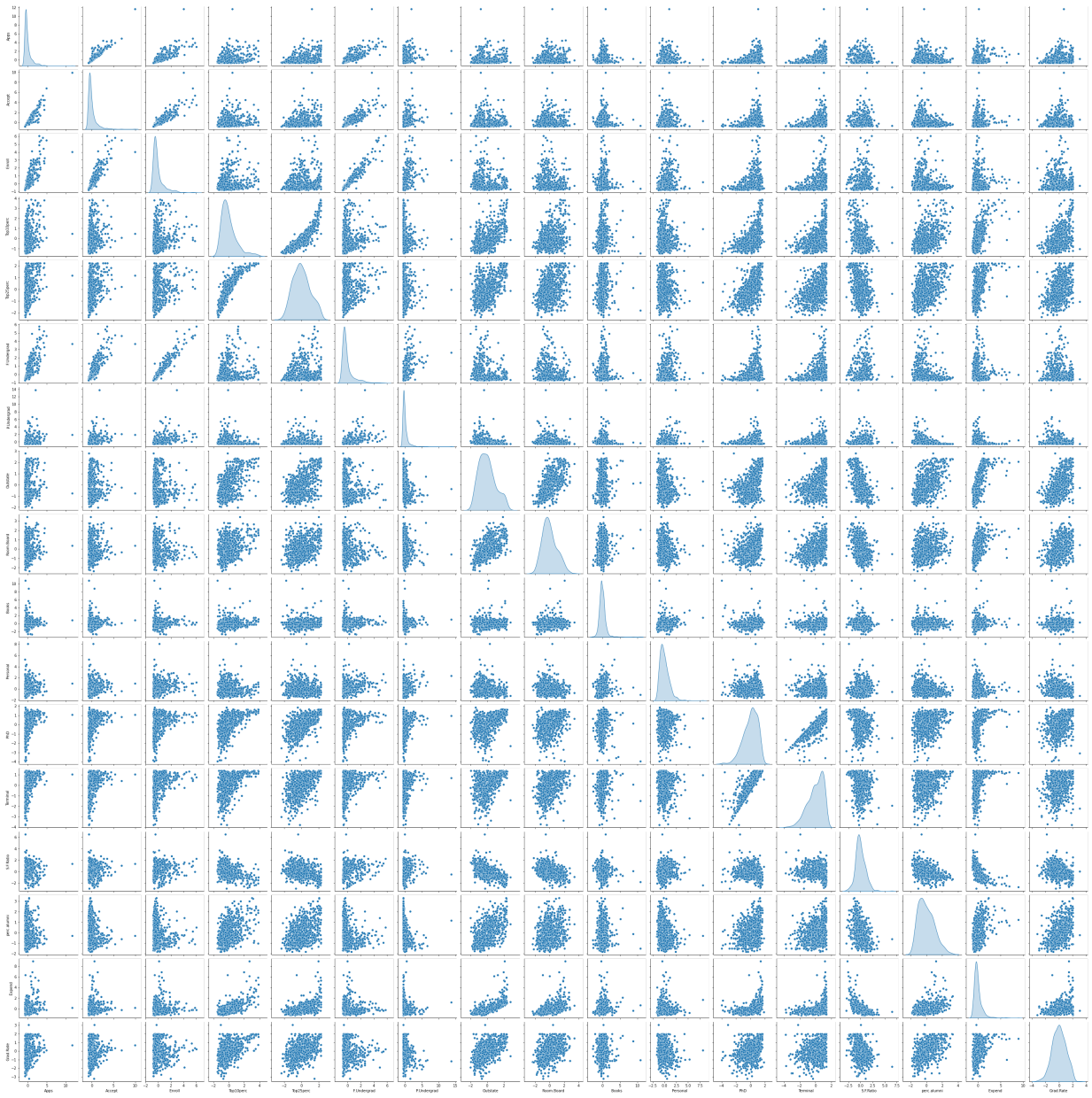
	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board
0	1660	1232	721	23	52	2885	537	7440	3300
1	2186	1924	512	16	29	2683	1227	12280	6450
2	1428	1097	336	22	50	1036	99	11250	3750
3	417	349	137	60	89	510	63	12960	5450
4	193	146	55	16	44	249	869	7560	4120

In [131]:
```
for x in listNumericColumns:
    univariateAnalysis_numeric(x,100)
```

```
BoxPlot of Apps
--------------------------------------------------------------------------------
```

EDA: Bivariate Analysis

In [200]:
```
# Pairplot comprises of scatter plots between different columns & histogram betwe
sns.pairplot(df_clg,diag_kind='kde');
```



In the above plot scatter diagrams are plotted for all the numerical columns in the dataset. A scatter plot is a visual representation of the degree of correlation between any two columns. The pair plot function in seaborn makes it very easy to generate joint scatter plots for all the columns in the data.

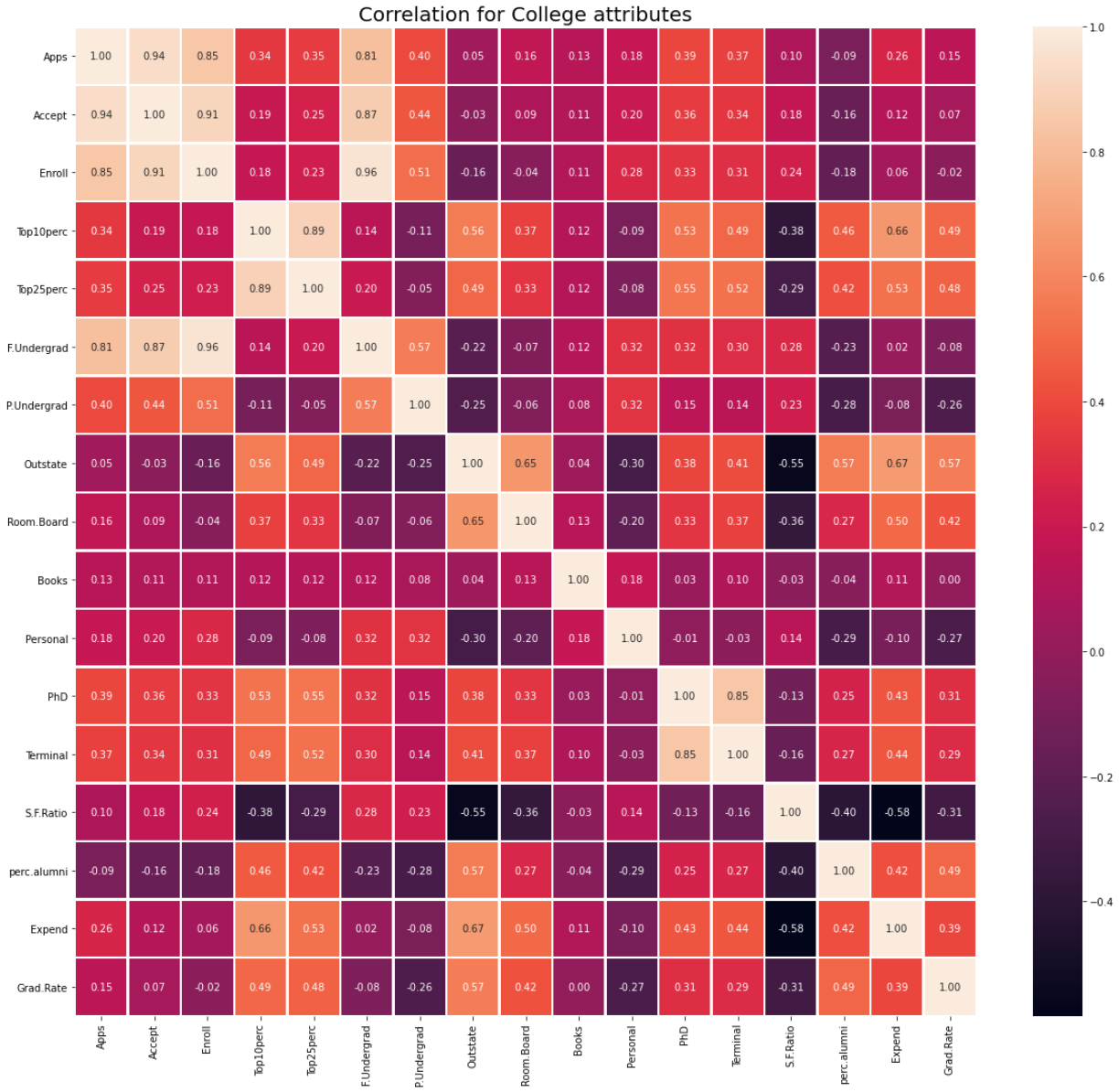
In [133]:
```
# np.random.seed(1234)
# data = np.random.rand(800,2)
# df = pd.DataFrame(data=data, columns=['x','y'])

# ## split the dataframe into 17 chunks, this is hardcoded since you specified th
# for index, df_chunk in enumerate(np.array_split(df, 54)):
#     plt.scatter(df_chunk.x, df_chunk.y)

#     ## this will immediately save 54 scatter plots, numbered 1-54, be warned!
#     plt.title('Scatter Plot #' + str(index+1))
#     plt.savefig('./scatter_plot_' + str(index+1) + '.png')

#     ## clear the figure
#     plt.clf()
```

In [134]:
```
df_clg.corr(method='pearson')
```

Out[134]:

	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outsta
Apps	1.000000	0.943451	0.846822	0.338834	0.351640	0.814491	0.398264	0.05015
Accept	0.943451	1.000000	0.911637	0.192447	0.247476	0.874223	0.441271	-0.02575
Enroll	0.846822	0.911637	1.000000	0.181294	0.226745	0.964640	0.513069	-0.1554
Top10perc	0.338834	0.192447	0.181294	1.000000	0.891995	0.141289	-0.105356	0.5623
Top25perc	0.351640	0.247476	0.226745	0.891995	1.000000	0.199445	-0.053577	0.4893
Undergrad	0.814491	0.874223	0.964640	0.141289	0.199445	1.000000	0.570512	-0.2157
Undergrad	0.398264	0.441271	0.513069	-0.105356	-0.053577	0.570512	1.000000	-0.2535
Outstate	0.050159	-0.025755	-0.155477	0.562331	0.489394	-0.215742	-0.253512	1.00000
oom.Board	0.164939	0.090899	-0.040232	0.371480	0.331490	-0.068890	-0.061326	0.6542
Books	0.132559	0.113525	0.112711	0.118858	0.115527	0.115550	0.081200	0.0388
Personal	0.178731	0.200989	0.280929	-0.093316	-0.080810	0.317200	0.319882	-0.2990
PhD	0.390697	0.355758	0.331469	0.531828	0.545862	0.318337	0.149114	0.3829
Terminal	0.369491	0.337583	0.308274	0.491135	0.524749	0.300019	0.141904	0.4079
S.F.Ratio	0.095633	0.176229	0.237271	-0.384875	-0.294629	0.279703	0.232531	-0.5548
erc.alumni	-0.090226	-0.159990	-0.180794	0.455485	0.417864	-0.229462	-0.280792	0.5662
Expend	0.259592	0.124717	0.064169	0.660913	0.527447	0.018652	-0.083568	0.6727
Grad.Rate	0.146755	0.067313	-0.022341	0.494989	0.477281	-0.078773	-0.257001	0.5712

In [135]:
```
f,ax =plt.subplots(figsize=(20,18))
sns.heatmap(corr, annot=True , linewidths=1.5, fmt ='.2f', ax=ax)
plt.title('Correlation for College attributes', fontsize=20)
plt.show()
```



In [137]:
```
# All variables are not on same scale, Hence, we have to perform scaling here
```

In [138]:
```
from sklearn.preprocessing import StandardScaler
std_scale = StandardScaler()
std_scale
```

Out[138]:
```
StandardScaler()
```

In [139]:
```
cols = list(df_num.columns.values)
len(cols)
```

Out[139]:
```
17
```

In [140]:
```
for i in range(len(cols)):
    df_clg[cols[i]] = std_scale.fit_transform(df_clg[[cols[i]]])

# df_num['Apps'] = std_scale.fit_transform(df_num[['Apps']])
# df_num['Accept'] = std_scale.fit_transform(df_num[['Accept']])
# df_num['Enroll'] = std_scale.fit_transform(df_num[['Enroll']])
# df_num['Top10perc'] = std_scale.fit_transform(df_num[['Top10perc']])
# df_num['Top25perc'] = std_scale.fit_transform(df_num[['Top25perc']])
# df_num['F.Undergrad'] = std_scale.fit_transform([['F.Undergrad']])
# df_num['P.Undergrad'] = std_scale.fit_transform([['P.Undergrad']])
# df_num['Outstate'] = std_scale.fit_transform([['Outstate']])
# df_num['Room.Board'] = std_scale.fit_transform([['Room.Board']])
# df_num['Books'] = std_scale.fit_transform([['Books']])
# df_num['Personal'] = std_scale.fit_transform([['Personal']])
# df_num['PhD'] = std_scale.fit_transform([['PhD']])
# df_num['Terminal'] = std_scale.fit_transform([['Terminal']])
# df_num['S.F.Ratio'] = std_scale.fit_transform([['S.F.Ratio']])
# df_num['perc.alumni'] = std_scale.fit_transform([['perc.alumni']])
# df_num['Expend'] = std_scale.fit_transform([['Expend']])
# df_num['Grad.Rate'] = std_scale.fit_transform([['Grad.Rate']])
```

In [141]:
```
print("SCALED DATASET USING STANDARD SCALER")
df_clg.head().T
```

```
SCALED DATASET USING STANDARD SCALER
```

Out[141]:

	0	1	2	3	4
Names	Abilene Christian University	Adelphi University	Adrian College	Agnes Scott College	Alaska Pacific University
Apps	-0.346882	-0.210884	-0.406866	-0.668261	-0.726176
Accept	-0.321205	-0.038703	-0.376318	-0.681682	-0.764555
Enroll	-0.063509	-0.288584	-0.478121	-0.692427	-0.780735
Top10perc	-0.258583	-0.655656	-0.315307	1.840231	-0.655656
Top25perc	-0.191827	-1.353911	-0.292878	1.677612	-0.596031
F.Undergrad	-0.168116	-0.209788	-0.549565	-0.658079	-0.711924
P.Undergrad	-0.209207	0.244307	-0.49709	-0.520752	0.009005
Outstate	-0.746356	0.457496	0.201305	0.626633	-0.716508
Room.Board	-0.964905	1.909208	-0.554317	0.996791	-0.216723
Books	-0.602312	1.21588	-0.905344	-0.602312	1.518912
Personal	1.270045	0.235515	-0.259582	-0.688173	0.235515
PhD	-0.163028	-2.675646	-1.204845	1.185206	0.204672
Terminal	-0.115729	-3.378176	-0.931341	1.175657	-0.523535
S.F.Ratio	1.013776	-0.477704	-0.300749	-1.615274	-0.553542
perc.alumni	-0.867574	-0.544572	0.585935	1.151188	-1.675079
Expend	-0.50191	0.16611	-0.17729	1.792851	0.241803
Grad.Rate	-0.318252	-0.551262	-0.667767	-0.376504	-2.939613

If you look at the variables , all of them have been normalized and scaled in one scale now.

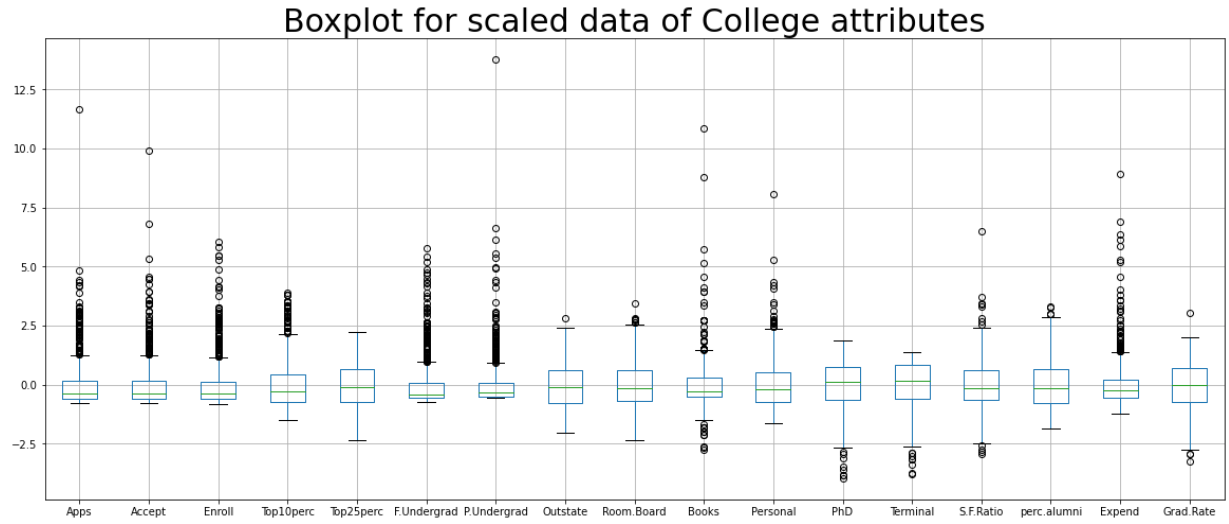
In [181]:
```
from scipy.stats import zscore
df_num_scaled = df_num.apply(zscore)
print("SCALED DATASET USING ZSCORE")
df_num_scaled.head().T
```

```
SCALED DATASET USING ZSCORE
```

Out[181]:

	0	1	2	3	4
Apps	-0.346882	-0.210884	-0.406866	-0.668261	-0.726176
Accept	-0.321205	-0.038703	-0.376318	-0.681682	-0.764555
Enroll	-0.063509	-0.288584	-0.478121	-0.692427	-0.780735
Top10perc	-0.258583	-0.655656	-0.315307	1.840231	-0.655656
Top25perc	-0.191827	-1.353911	-0.292878	1.677612	-0.596031
F.Undergrad	-0.168116	-0.209788	-0.549565	-0.658079	-0.711924
P.Undergrad	-0.209207	0.244307	-0.497090	-0.520752	0.009005
Outstate	-0.746356	0.457496	0.201305	0.626633	-0.716508
Room.Board	-0.964905	1.909208	-0.554317	0.996791	-0.216723
Books	-0.602312	1.215880	-0.905344	-0.602312	1.518912
Personal	1.270045	0.235515	-0.259582	-0.688173	0.235515
PhD	-0.163028	-2.675646	-1.204845	1.185206	0.204672
Terminal	-0.115729	-3.378176	-0.931341	1.175657	-0.523535
S.F.Ratio	1.013776	-0.477704	-0.300749	-1.615274	-0.553542
perc.alumni	-0.867574	-0.544572	0.585935	1.151188	-1.675079
Expend	-0.501910	0.166110	-0.177290	1.792851	0.241803
Grad.Rate	-0.318252	-0.551262	-0.667767	-0.376504	-2.939613

In [170]:
```
df_new.boxplot(figsize=(20,8));
plt.title('Boxplot for scaled data of College attributes', fontsize=30)
plt.show()
```



Create a covariance matrix for identifying Principal components

In [182]:
```
# Step 1 - Create covariance matrix
cov_matrix = np.cov(df_num_scaled.T)
print('Covariance matrix : \n\n' , cov_matrix)
```

```
Covariance matrix :

 [[ 1.00128866  0.94466636  0.84791332  0.33927032  0.35209304  0.81554018
   0.3987775   0.05022367  0.16515151  0.13272942  0.17896117  0.39120081
   0.36996762  0.09575627 -0.09034216  0.2599265   0.14694372]
 [ 0.94466636  1.00128866  0.91281145  0.19269493  0.24779465  0.87534985
   0.44183938 -0.02578774  0.09101577  0.11367165  0.20124767  0.35621633
   0.3380184   0.17645611 -0.16019604  0.12487773  0.06739929]
 [ 0.84791332  0.91281145  1.00128866  0.18152715  0.2270373   0.96588274
   0.51372977 -0.1556777  -0.04028353  0.11285614  0.28129148  0.33189629
   0.30867133  0.23757707 -0.18102711  0.06425192 -0.02236983]
 [ 0.33927032  0.19269493  0.18152715  1.00128866  0.89314445  0.1414708
  -0.10549205  0.5630552   0.37195909  0.1190116  -0.09343665  0.53251337
   0.49176793 -0.38537048  0.45607223  0.6617651   0.49562711]
 [ 0.35209304  0.24779465  0.2270373   0.89314445  1.00128866  0.19970167
  -0.05364569  0.49002449  0.33191707  0.115676   -0.08091441  0.54656564
   0.52542506 -0.29500852  0.41840277  0.52812713  0.47789622]
 [ 0.81554018  0.87534985  0.96588274  0.1414708   0.19970167  1.00128866
   0.57124738 -0.21602002 -0.06897917  0.11569867  0.31760831  0.3187472
   0.30040557  0.28006379 -0.22975792  0.01867565 -0.07887464]
 [ 0.3987775   0.44183938  0.51372977 -0.10549205 -0.05364569  0.57124738
   1.00128866 -0.25383901 -0.06140453  0.08130416  0.32029384  0.14930637
   0.14208644  0.23283016 -0.28115421 -0.08367612 -0.25733218]
 [ 0.05022367 -0.02578774 -0.1556777   0.5630552   0.49002449 -0.21602002
  -0.25383901  1.00128866  0.65509951  0.03890494 -0.29947232  0.38347594
   0.40850895 -0.55553625  0.56699214  0.6736456   0.57202613]
 [ 0.16515151  0.09101577 -0.04028353  0.37195909  0.33191707 -0.06897917
  -0.06140453  0.65509951  1.00128866  0.12812787 -0.19968518  0.32962651
   0.3750222  -0.36309504  0.27271444  0.50238599  0.42548915]
 [ 0.13272942  0.11367165  0.11285614  0.1190116   0.115676    0.11569867
   0.08130416  0.03890494  0.12812787  1.00128866  0.17952581  0.0269404
   0.10008351 -0.03197042 -0.04025955  0.11255393  0.00106226]
 [ 0.17896117  0.20124767  0.28129148 -0.09343665 -0.08091441  0.31760831
   0.32029384 -0.29947232 -0.19968518  0.17952581  1.00128866 -0.01094989
  -0.03065256  0.13652054 -0.2863366  -0.09801804 -0.26969106]
 [ 0.39120081  0.35621633  0.33189629  0.53251337  0.54656564  0.3187472
   0.14930637  0.38347594  0.32962651  0.0269404  -0.01094989  1.00128866
   0.85068186 -0.13069832  0.24932955  0.43331936  0.30543094]
 [ 0.36996762  0.3380184   0.30867133  0.49176793  0.52542506  0.30040557
   0.14208644  0.40850895  0.3750222   0.10008351 -0.03065256  0.85068186
   1.00128866 -0.16031027  0.26747453  0.43936469  0.28990033]
 [ 0.09575627  0.17645611  0.23757707 -0.38537048 -0.29500852  0.28006379
   0.23283016 -0.55553625 -0.36309504 -0.03197042  0.13652054 -0.13069832
  -0.16031027  1.00128866 -0.4034484  -0.5845844  -0.30710565]
 [-0.09034216 -0.16019604 -0.18102711  0.45607223  0.41840277 -0.22975792
  -0.28115421  0.56699214  0.27271444 -0.04025955 -0.2863366   0.24932955
   0.26747453 -0.4034484   1.00128866  0.41825001  0.49153016]
 [ 0.2599265   0.12487773  0.06425192  0.6617651   0.52812713  0.01867565
  -0.08367612  0.6736456   0.50238599  0.11255393 -0.09801804  0.43331936
   0.43936469 -0.5845844   0.41825001  1.00128866  0.39084571]
 [ 0.14694372  0.06739929 -0.02236983  0.49562711  0.47789622 -0.07887464
  -0.25733218  0.57202613  0.42548915  0.00106226 -0.26969106  0.30543094
   0.28990033 -0.30710565  0.49153016  0.39084571  1.00128866]]
```

In [183]:
```
# Step 2- Get eigen values and eigen vector
eig_vals , eig_vecs = np.linalg.eig(cov_matrix)
print('EIGEN VECTORS : \n\n', eig_vecs)
print('EIGEN VALUES : \n\n' , eig_vals)
```

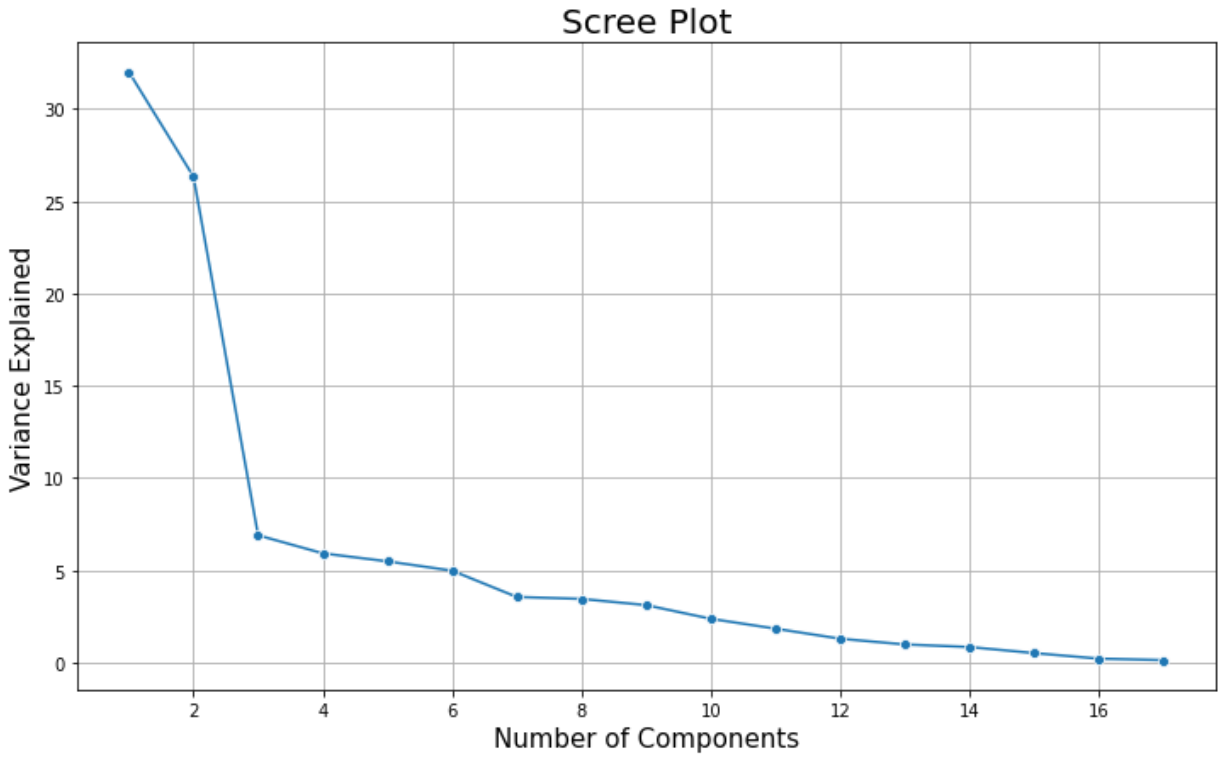
```
EIGEN VECTORS :

 [[-2.48765602e-01  3.31598227e-01  6.30921033e-02 -2.81310530e-01
   5.74140964e-03  1.62374420e-02  4.24863486e-02  1.03090398e-01
   9.02270802e-02 -5.25098025e-02  3.58970400e-01 -4.59139498e-01
   4.30462074e-02 -1.33405806e-01  8.06328039e-02 -5.95830975e-01
   2.40709086e-02]
 [-2.07601502e-01  3.72116750e-01  1.01249056e-01 -2.67817346e-01
   5.57860920e-02 -7.53468452e-03  1.29497196e-02  5.62709623e-02
   1.77864814e-01 -4.11400844e-02 -5.43427250e-01  5.18568789e-01
  -5.84055850e-02  1.45497511e-01  3.34674281e-02 -2.92642398e-01
  -1.45102446e-01]
 [-1.76303592e-01  4.03724252e-01  8.29855709e-02 -1.61826771e-01
  -5.56936353e-02  4.25579803e-02  2.76928937e-02 -5.86623552e-02
   1.28560713e-01 -3.44879147e-02  6.09651110e-01  4.04318439e-01
  -6.93988831e-02 -2.95896092e-02 -8.56967180e-02  4.44638207e-01
   1.11431545e-02]
 [-3.54273947e-01 -8.24118211e-02 -3.50555339e-02  5.15472524e-02
  -3.95434345e-01  5.26927980e-02  1.61332069e-01  1.22678028e-01
  -3.41099863e-01 -6.40257785e-02 -1.44986329e-01  1.48738723e-01
  -8.10481404e-03 -6.97722522e-01 -1.07828189e-01 -1.02303616e-03
   3.85543001e-02]
 [-3.44001279e-01 -4.47786551e-02  2.41479376e-02  1.09766541e-01
  -4.26533594e-01 -3.30915896e-02  1.18485556e-01  1.02491967e-01
  -4.03711989e-01 -1.45492289e-02  8.03478445e-02 -5.18683400e-02
  -2.73128469e-01  6.17274818e-01  1.51742110e-01 -2.18838802e-02
  -8.93515563e-02]
 [-1.54640962e-01  4.17673774e-01  6.13929764e-02 -1.00412335e-01
  -4.34543659e-02  4.34542349e-02  2.50763629e-02 -7.88896442e-02
   5.94419181e-02 -2.08471834e-02 -4.14705279e-01 -5.60363054e-01
  -8.11578181e-02 -9.91640992e-03 -5.63728817e-02  5.23622267e-01
   5.61767721e-02]
 [-2.64425045e-02  3.15087830e-01 -1.39681716e-01  1.58558487e-01
   3.02385408e-01  1.91198583e-01 -6.10423460e-02 -5.70783816e-01
  -5.60672902e-01  2.23105808e-01  9.01788964e-03  5.27313042e-02
   1.00693324e-01 -2.09515982e-02  1.92857500e-02 -1.25997650e-01
  -6.35360730e-02]
 [-2.94736419e-01 -2.49643522e-01 -4.65988731e-02 -1.31291364e-01
   2.22532003e-01  3.00003910e-02 -1.08528966e-01 -9.84599754e-03
   4.57332880e-03 -1.86675363e-01  5.08995918e-02 -1.01594830e-01
   1.43220673e-01 -3.83544794e-02 -3.40115407e-02  1.41856014e-01
  -8.23443779e-01]
 [-2.49030449e-01 -1.37808883e-01 -1.48967389e-01 -1.84995991e-01
   5.60919470e-01 -1.62755446e-01 -2.09744235e-01  2.21453442e-01
  -2.75022548e-01 -2.98324237e-01  1.14639620e-03  2.59293381e-02
  -3.59321731e-01 -3.40197083e-03 -5.84289756e-02  6.97485854e-02
   3.54559731e-01]
 [-6.47575181e-02  5.63418434e-02 -6.77411649e-01 -8.70892205e-02
  -1.27288825e-01 -6.41054950e-01  1.49692034e-01 -2.13293009e-01
   1.33663353e-01  8.20292186e-02  7.72631963e-04 -2.88282896e-03
   3.19400370e-02  9.43887925e-03 -6.68494643e-02 -1.14379958e-02
  -2.81593679e-02]
```

```
 [ 4.25285386e-02  2.19929218e-01 -4.99721120e-01  2.30710568e-01
  -2.22311021e-01  3.31398003e-01 -6.33790064e-01  2.32660840e-01
   9.44688900e-02 -1.36027616e-01 -1.11433396e-03  1.28904022e-02
  -1.85784733e-02  3.09001353e-03  2.75286207e-02 -3.94547417e-02
  -3.92640266e-02]
 [-3.18312875e-01  5.83113174e-02  1.27028371e-01  5.34724832e-01
   1.40166326e-01 -9.12555212e-02  1.09641298e-03  7.70400002e-02
   1.85181525e-01  1.23452200e-01  1.38133366e-02 -2.98075465e-02
   4.03723253e-02  1.12055599e-01 -6.91126145e-01 -1.27696382e-01
   2.32224316e-02]
 [-3.17056016e-01  4.64294477e-02  6.60375454e-02  5.19443019e-01
   2.04719730e-01 -1.54927646e-01  2.84770105e-02  1.21613297e-02
   2.54938198e-01  8.85784627e-02  6.20932749e-03  2.70759809e-02
  -5.89734026e-02 -1.58909651e-01  6.71008607e-01  5.83134662e-02
   1.64850420e-02]
 [ 1.76957895e-01  2.46665277e-01  2.89848401e-01  1.61189487e-01
  -7.93882496e-02 -4.87045875e-01 -2.19259358e-01  8.36048735e-02
  -2.74544380e-01 -4.72045249e-01 -2.22215182e-03  2.12476294e-02
   4.45000727e-01  2.08991284e-02  4.13740967e-02  1.77152700e-02
  -1.10262122e-02]
 [-2.05082369e-01 -2.46595274e-01  1.46989274e-01 -1.73142230e-02
  -2.16297411e-01  4.73400144e-02 -2.43321156e-01 -6.78523654e-01
   2.55334907e-01 -4.22999706e-01 -1.91869743e-02 -3.33406243e-03
  -1.30727978e-01  8.41789410e-03 -2.71542091e-02 -1.04088088e-01
   1.82660654e-01]
 [-3.18908750e-01 -1.31689865e-01 -2.26743985e-01 -7.92734946e-02
   7.59581203e-02  2.98118619e-01  2.26584481e-01  5.41593771e-02
   4.91388809e-02 -1.32286331e-01 -3.53098218e-02  4.38803230e-02
   6.92088870e-01  2.27742017e-01  7.31225166e-02  9.37464497e-02
   3.25982295e-01]
 [-2.52315654e-01 -1.69240532e-01  2.08064649e-01 -2.69129066e-01
  -1.09267913e-01 -2.16163313e-01 -5.59943937e-01  5.33553891e-03
  -4.19043052e-02  5.90271067e-01 -1.30710024e-02  5.00844705e-03
   2.19839000e-01  3.39433604e-03  3.64767385e-02  6.91969778e-02
   1.22106697e-01]]
EIGEN VALUES :

 [5.45052162 4.48360686 1.17466761 1.00820573 0.93423123 0.84849117
 0.6057878  0.58787222 0.53061262 0.4043029  0.02302787 0.03672545
 0.31344588 0.08802464 0.1439785  0.16779415 0.22061096]
```

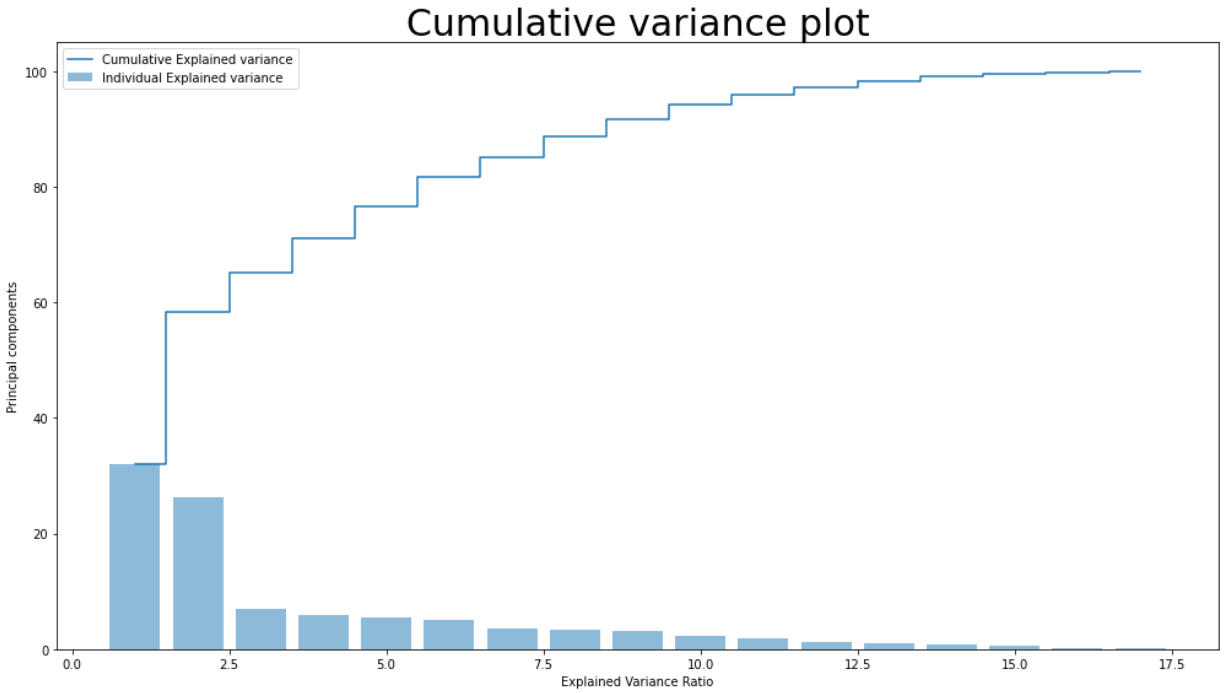
In [174]:
```
#  Find variance & cumulative variance
tot = sum(eig_vals)
var_exp = [(i/tot) *100 for i in sorted(eig_vals, reverse=True)]
cum_var_exp = np.cumsum(var_exp)
print('Cumulative variable explained :', cum_var_exp)
```

```
Cumulative variable explained : [ 32.0206282   58.36084263  65.26175919  71.184
74841  76.67315352
  81.65785448  85.21672597  88.67034731  91.78758099  94.16277251
  96.00419883  97.30024023  98.28599436  99.13183669  99.64896227
  99.86471628 100.        ]
```

In [175]:
```
# Step 3 : View Scree Plot to identify the number of components to be built
plt.figure(figsize=(12,7))
sns.lineplot(y=var_exp,x=range(1,len(var_exp)+1),marker='o')
plt.xlabel('Number of Components',fontsize=15)
plt.ylabel('Variance Explained',fontsize=15)
plt.title('Scree Plot',fontsize=20)
plt.grid()
plt.show()
```



In [176]:
```
# plotting
plt.figure(figsize = (14,8))
plt.bar(range(1,eig_vals.size+1), var_exp, alpha= 0.5,align='center', label='Indi
plt.step(range(1,eig_vals.size+1),cum_var_exp, where ='mid', label='Cumulative Ex
plt.title('Cumulative variance plot', fontsize=30)
plt.xlabel('Explained Variance Ratio')
plt.ylabel('Principal components')
plt.legend(loc='best')
plt.tight_layout()
plt.show()
```



In [189]:
```
# Step 4 : Apply PCA for the number of decided components to get the loadings and
# Using scikit learn PCA here. It does all the above steps and maps data to PCA d
from sklearn.decomposition import PCA
# NOTE - we are generating only 6 PCA dimensions (dimensionality reduction from 1
pca = PCA(n_components=6, random_state=123)
df_pca = pca.fit_transform(df_num_scaled)
df_pca.transpose() # Component output
```

Out[189]:
```
array([[-1.59285540e+00, -2.19240180e+00, -1.43096371e+00, ...,
        -7.32560596e-01,  7.91932735e+00, -4.69508066e-01],
       [ 7.67333510e-01, -5.78829984e-01, -1.09281889e+00, ...,
        -7.72352401e-02, -2.06832886e+00,  3.66660943e-01],
       [-1.01073616e-01,  2.27879810e+00, -4.38092815e-01, ...,
        -4.05798710e-04,  2.07356387e+00, -1.32891523e+00],
       [-9.21749291e-01,  3.58891825e+00,  6.77240533e-01, ...,
         5.43164956e-02,  8.52053749e-01, -1.08022442e-01],
       [-7.43975435e-01,  1.05999660e+00, -3.69613276e-01, ...,
        -5.16021192e-01, -9.47754660e-01, -1.13217598e+00],
       [-2.98306092e-01, -1.77137311e-01, -9.60591689e-01, ...,
         4.68014225e-01, -2.06993735e+00,  8.39893075e-01]])
```

In [190]:
```
# Loading of each feature on the components, these are the same as the Eigen vect
pca.components_
```

Out[190]:
```
array([[ 0.2487656 ,  0.2076015 ,  0.17630359,  0.35427395,  0.34400128,
         0.15464096,  0.0264425 ,  0.29473642,  0.24903045,  0.06475752,
        -0.04252854,  0.31831287,  0.31705602, -0.17695789,  0.20508237,
         0.31890875,  0.25231565],
       [ 0.33159823,  0.37211675,  0.40372425, -0.08241182, -0.04477866,
         0.41767377,  0.31508783, -0.24964352, -0.13780888,  0.05634184,
         0.21992922,  0.05831132,  0.04642945,  0.24666528, -0.24659527,
        -0.13168986, -0.16924053],
       [-0.06309209, -0.10124907, -0.08298558,  0.03505553, -0.02414794,
        -0.06139296,  0.13968171,  0.04659888,  0.14896739,  0.67741165,
         0.49972112, -0.12702837, -0.06603755, -0.2898484 , -0.14698927,
         0.22674398, -0.20806465],
       [ 0.28131052,  0.26781736,  0.16182679, -0.05154725, -0.10976654,
         0.10041231, -0.15855849,  0.13129136,  0.18499599,  0.08708922,
        -0.23071057, -0.53472483, -0.51944302, -0.16118949,  0.01731422,
         0.0792735 ,  0.26912907],
       [ 0.00574142,  0.05578609, -0.05569364, -0.39543435, -0.42653359,
        -0.04345436,  0.30238541,  0.222532  ,  0.56091947, -0.12728883,
        -0.22231102,  0.14016633,  0.20471973, -0.07938825, -0.21629741,
```

In [191]:
```
pca.explained_variance_ratio_
```

Out[191]:
```
array([0.32020628, 0.26340214, 0.06900917, 0.05922989, 0.05488405,
       0.04984701])
```

In [192]:
```
# Create a dataframe of component loading against each field to identify the patt

df_pca_loading = pd.DataFrame(pca.components_,columns=list(df_num_scaled))
df_pca_loading.shape
```

Out[192]:
```
(6, 17)
```

In [196]:
```
df_pca_loading.T
```

Out[196]:

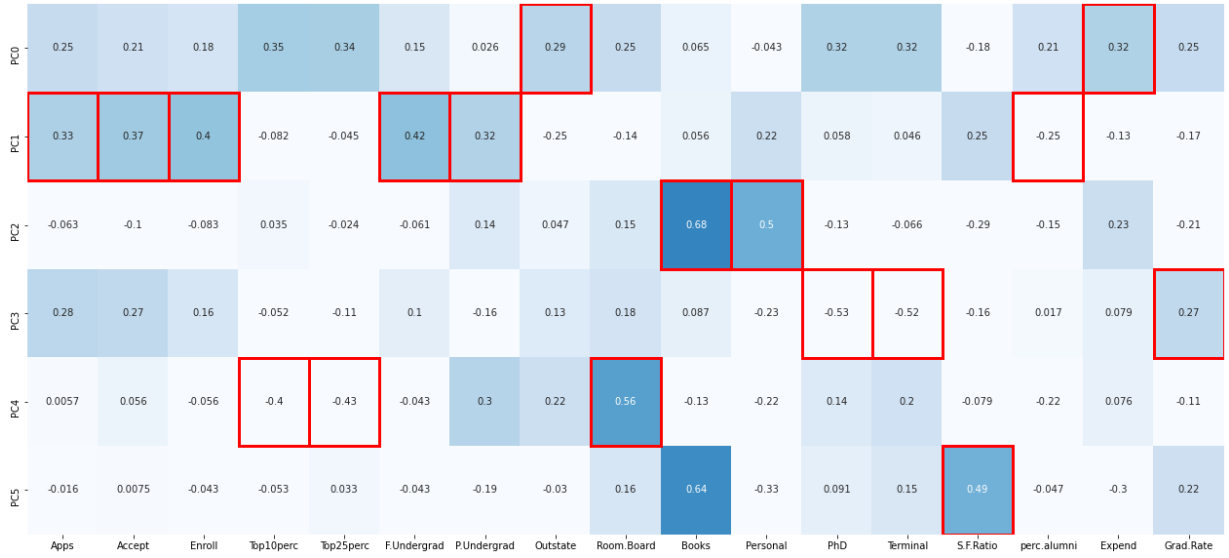
	0	1	2	3	4	5
Apps	0.248766	0.331598	-0.063092	0.281311	0.005741	-0.016237
Accept	0.207602	0.372117	-0.101249	0.267817	0.055786	0.007535
Enroll	0.176304	0.403724	-0.082986	0.161827	-0.055694	-0.042558
Top10perc	0.354274	-0.082412	0.035056	-0.051547	-0.395434	-0.052693
Top25perc	0.344001	-0.044779	-0.024148	-0.109767	-0.426534	0.033092
F.Undergrad	0.154641	0.417674	-0.061393	0.100412	-0.043454	-0.043454
P.Undergrad	0.026443	0.315088	0.139682	-0.158558	0.302385	-0.191199
Outstate	0.294736	-0.249644	0.046599	0.131291	0.222532	-0.030000
Room.Board	0.249030	-0.137809	0.148967	0.184996	0.560919	0.162755
Books	0.064758	0.056342	0.677412	0.087089	-0.127289	0.641055
Personal	-0.042529	0.219929	0.499721	-0.230711	-0.222311	-0.331398
PhD	0.318313	0.058311	-0.127028	-0.534725	0.140166	0.091256
Terminal	0.317056	0.046429	-0.066038	-0.519443	0.204720	0.154928
S.F.Ratio	-0.176958	0.246665	-0.289848	-0.161189	-0.079388	0.487046
perc.alumni	0.205082	-0.246595	-0.146989	0.017314	-0.216297	-0.047340
Expend	0.318909	-0.131690	0.226744	0.079273	0.075958	-0.298119
Grad.Rate	0.252316	-0.169241	-0.208065	0.269129	-0.109268	0.216163

Let's identify which features have maximum loading across the components.

- We will first plot the component loading on a heatmap.
- For each feature, we find the maximum loading value across the components and mark the same with help of rectangular box.
- Features marked with rectangular red box are the one having maximum loading on the respective component. We consider these marked features to decide the context that the component represents

In [197]:
```
from matplotlib.patches import Rectangle
```

In [216]:
```
# To find out which variable has the highest coefficient across the PC's
fig,ax = plt.subplots(figsize=(22, 10), facecolor='w', edgecolor='k')
ax = sns.heatmap(df_pca_loading, annot=True, vmax=1.0, vmin=0, cmap='Blues', cbar
                 yticklabels=['PC0','PC1','PC2','PC3','PC4','PC5'])

column_max = df_pca_loading.abs().idxmax(axis=0)

for col, variable in enumerate(df_pca_loading.columns):
    position = df_pca_loading.index.get_loc(column_max[variable])
    ax.add_patch(Rectangle((col, position),1,1, fill=False, edgecolor='red', lw=3
```



In [199]:
```
df_pca = pd.DataFrame(df_pca,columns=['PC_Expenses_Of_Outstates' , 'PC_College_Fu
```

In [203]:
```
df_pca.head().T
```

Out[203]:

	0	1	2	3	
PC_Expenses_Of_Outstates	-1.592855	-2.192402	-1.430964	2.855557	-2.212
PC_College_Funds_Collected_From_Students	0.767334	-0.578830	-1.092819	-2.630612	0.021
PC_Student_Expenses	-0.101074	2.278798	-0.438093	0.141722	2.387
PC_Graduates_Or_More	-0.921749	3.588918	0.677241	-1.295486	-1.114
PC_Accomodation_Expenses_Incurred_On_Toppers	-0.743975	1.059997	-0.369613	-0.183837	0.684
PC_Student_By_Faculty_Ratio	-0.298306	-0.177137	-0.960592	-1.059508	0.004

In [204]:
```
df_pca.shape
```

Out[204]: (777, 6)

In [205]:
```
data_new = pd.concat([df_clg['Names'], df_pca], axis=1)
```

In [206]:
```
data_new.shape
```

Out[206]: (777, 7)

In [210]:
```
print("NEW DATASET - POST DIMENSION REDUCTION")
data_new.head().T
```

```
NEW DATASET - POST DIMENSION REDUCTION
```

Out[210]:

	0	1	2	3	
Names	Abilene Christian University	Adelphi University	Adrian College	Agnes Scott College	Ala Pa Unive
PC_Expenses_Of_Outstates	-1.592855	-2.192402	-1.430964	2.855557	-2.212
PC_College_Funds_Collected_From_Students	0.767334	-0.57883	-1.092819	-2.630612	0.021
PC_Student_Expenses	-0.101074	2.278798	-0.438093	0.141722	2.38
PC_Graduates_Or_More	-0.921749	3.588918	0.677241	-1.295486	-1.114
PC_Accomodation_Expenses_Incurred_On_Toppers	-0.743975	1.059997	-0.369613	-0.183837	0.684
PC_Student_By_Faculty_Ratio	-0.298306	-0.177137	-0.960592	-1.059508	0.004

In [222]:
```
print('DESCRIPTIVE SUMMARY OF NEW DATASET')
df_pca.describe(include='all').T
```

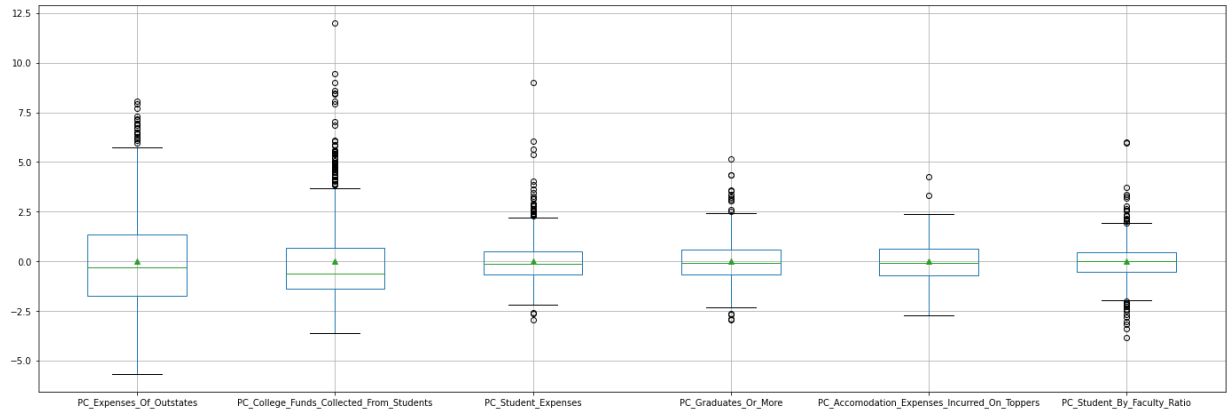
DESCRIPTIVE SUMMARY OF NEW DATASET

Out[222]:

	count	mean	std	min	25%
PC_Expenses_Of_Outstates	777.0	4.693800e-17	2.334635	-5.662905	-1.73120
PC_College_Funds_Collected_From_Students	777.0	5.701145e-17	2.117453	-3.590891	-1.34807
PC_Student_Expenses	777.0	2.286174e-18	1.083821	-2.941286	-0.66630
PC_Graduates_Or_More	777.0	-9.267933e-17	1.004094	-2.943103	-0.65583
PC_Accomodation_Expenses_Incurred_On_Toppers	777.0	8.573151e-18	0.966556	-2.690124	-0.69985
PC_Student_By_Faculty_Ratio	777.0	-1.100221e-17	0.921136	-3.822954	-0.52295

In [225]:
```
print('BOXPLOT FOR NEW DATASET WITH REDUCED DIMENSIONS')
df_pca.boxplot(showmeans =True, figsize=(24,8));
```

BOXPLOT FOR NEW DATASET WITH REDUCED DIMENSIONS



In [ ]: